TD2

Olivier Marchetti

octobre 2021

Exercice 1 : *Chaîne de caractères*

Voici une portion de code source *Java*.

```
public class TD2_exo1 {
    public static void main(String [] args){
        String uneChaine = "le java à Polytech, c'est Top";
        String unMessage = "le java à Polytech, c'est Top";
        String un = "le java à Polytech, ";
        String messageCoupe = "c'est Top";
        String messageColle = "le java à Polytech," + " c'est Top";
        String autosatisfaction = un + messageCoupe;
        String propagande = new String("le java à Polytech, c'est Top");
        String leitmotiv = new String("le java à Polytech, c'est Top");

        System.out.println("comparaison de uneChaine et unMessage : "
                                        + (uneChaine == unMessage));
        System.out.println("comparaison de uneChaine et messageColle : "
                                        + (uneChaine == messageColle));
        System.out.println("comparaison de uneChaine et autosatisfaction : "
                                        + (uneChaine == autosatisfaction));
        System.out.println("comparaison de uneChaine et propagande : "
                                        + (uneChaine == propagande));
        System.out.println("comparaison de propagande et leitmotiv : "
                                        + (propagande == leitmotiv));
    }
}
```

a) Sans taper le code source dans un éditeur de texte, prédire quel sera l'affichage produit à l'exécution du code.

b) Taper le code source et exécuter le code source et comparez avec vos prédictions. Déduisez en comment fonctionne le compilateur *Java* avec la classe *String*.

Exercice 2 : *Chaîne de caractères*

Ecrire un programme qui prend sur la ligne de commande un verbe du premier groupe et affiche la conjugaison de ce verbe au présent de l'indicatif (consultez les méthodes de la classe *String*).

Exercice 3 : *Chaîne de caractères*

Ecrice un programme qui prend en entrée sur la ligne de commandes un entier (compris en 0 et 999) et affiche ce même entier en toute lettre (consultez les méthodes de la classe *String*).

Exercice 4 : *Surcharge*

Voici un code source *Java* :

```
class Surcharge {
    int n;
    double x;

    public Surcharge() {
        n = 1;
        x = 3.5;
    }

    public Surcharge(int n, double x) {
        this.n = n;
        this.x = x;
    }

    int faireOperation(int p) {
        return 10 * p + n;
    }

    double faireOperation(double y, int p) {
        return x * p + y;
    }

    double faireOperation(int p, double y) {
        return ((double) n/p) + y;
    }
}

public class TD2_exo5 {
    public static void main (String [] args) {
        Surcharge s1 = new Surcharge();
        System.out.println(s1.faireOperation(2));
        System.out.println(s1.faireOperation(1.5, 4));
        System.out.println(s1.faireOperation(4, 1.5));
        s1 = new Surcharge(7, 2.0);
        System.out.println(s1.faireOperation(2));
    }
}
```

Donnez et expliquez l'affichage produit par l'exécution de ce code compilé.

Exercice 5 : *Héritage*

En respectant l'encapsulation des données, créer les classes suivantes :

- la classe `Personne` ayant comme attribut nom et age et comme méthodes deux constructeurs (l'un prenant ses arguments dans le code du main et l'autre initialisant ses attributs au clavier lors de l'exécution) et une méthode d'affichage.

- la classe `Scolaire` qui hérite de la classe Personne. Elle possède les attributs `nom_etablissement` et `niveau` (Primaire, Secondaire, Supérieur). On redéfinira la méthode d'affichage de la classe `Personne`.
- la classe `Etudiant` qui étend la classe `Scolaire`. Elle possède l'attribut `formation`. On redéfinira la méthode d'affichage.
- la classe `Lyceen` qui étend la classe `Scolaire`. Elle possède l'attribut `filiere_bac` (S,L,...). On redéfinira la méthode d'affichage.

Ecrire le code *Java* en proposant à chaque fois deux constructeurs de classe (un ordinaire qui prend ses arguments dans le main et un autre qui prend ses arguments venant du clavier lors de l'exécution).

Exercice 6 : *Héritage*

Soit le code *Java* suivant :

```
class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public static boolean sontIdentiques(Point a, Point b) {
        return ( (a.x == b.x) && (a.y == b.y) );
    }

    public boolean estIdentique(Point a) {
        return ( (a.x == x) && (a.y == y) );
    }
}

class PointNom extends Point {
    private char nom;

    public PointNom(int x, int y, char nom) {
        super(x,y);
        this.nom = nom;
    }
}

public class TD2_exo6 {
    public static void main (String [] args) {
        Point p = new Point(2,4);
        PointNom pn1 = new PointNom(2,4,'A');
        PointNom pn2 = new PointNom(2,4,'B');
        System.out.println(pn1.estIdentique(pn2));
        System.out.println(p.estIdentique(pn1));
        System.out.println(pn1.estIdentique(p));
        System.out.println(Point.sontIdentiques(pn1,pn2));
    }
}
```


a) Déroulez le code et explicitez les conversions et les appels de méthodes effectuées.

b) Redéfinissez les méthodes de la classe Point dans la classe PointNom de sorte à prendre en compte l'attribut de la classe PointNom. Reprendre la première question.

Exercice 7 : *Héritage*

Soit le code *Java* suivant :

```
class A {}

class B extends A {}

class TD2_exo9 {
    public static void main(String [] args) {
        Object monObjet = new B();
        System.out.println("monObjet est une instance de B : "
                           + (monObjet instanceof B));
        System.out.println("monObjet est une instance de A : "
                           + (monObjet instanceof A));
        System.out.println("monObjet est une instance de Object : "
                           + (monObjet instanceof Object));
        System.out.println("monObjet est une instance de String : "
                           + (monObjet instanceof String));
    }
}
```

Déterminez l'affichage produit par l'exécution de ce code compilé.

Exercice 8 : *Héritage*

a) Ecrire une classe Point2D qui modélise des points en deux dimensions ayant trois attributs encapsulés (x,y et norme). Les objets de cette classe disposeront d'une méthode d'affichage, d'une méthode de calcul de la norme et d'une méthode de calcul du produit scalaire.

b) A partir de la classe Point2D, proposez une classe Point3D pour les points en trois dimensions.